Technical Report 1653
June 1994

A Direct Decomposition Method for the Solution of Sparse Linear Least Squares Problems

A. K. Kevorkian

SUMMARY

Given a sparse nonsquare system of linear equations $Mx = b$ where M^TM is either dense or full, we present a direct method that generates a least squares solution of the original system $Mx = b$ by solving a smaller least squares problem. The method accomplishes this decomposition by applying orthogonal transformations to a restructured form of the original system of equations. The algorithms derived from the decomposition result are well-suited for both sequential and parallel architecture machines. In a specific Navy signal processing application, the presented algorithm computed on a Sun SPARC 10 workstation a least squares solution of a rank deficient system comprising 703 equations and 592 variables in a number of floating point computations tenfold smaller than a method that does not expoit the sparsity structure of M.

CONTENTS

Figures

Tables

1. INTRODUCTION

The method of least squares is a fundamental computational tool for estimating unknown parameters, curve fitting, data smoothing as well as solving nonsquare systems of equations. When the problem is formulated as a nonsquare system of linear equations

$$Mx = b \,, \tag{1}$$

in which M is an m by n full rank or rank deficient matrix, the linear least squares problem is expressed in the standard form

$$\min_{x} \| Mx - b \|_2 \ . \tag{2}$$

A number of efficient, direct methods for solving large and sparse linear least squares problems have been developed using orthogonalization, elimination and augmentation (George & Heath, 1980; Golub, 1965, Hachtel, 1974; Heath, 1982; Heath, 1984; Liu, 1986; Peters & Wilkinson, 1970). A central assumption in all these methods is that the matrix M^TM is sparse. In this work, we expand the choice of methods by developing a method that efficiently handles the case in which M is a sparse matrix while M^TM is dense, and possibly full. Our method is based on a decomposition result obtained from the application of orthogonal transformations to a restructured form of the system $Mx = b$. With this result, the original sparse linear least squares problem is reduced to a smaller least squares problem. The size of this smaller problem depends on the original zero-nonzero structure of the sparse matrix M. For a specific navy signal processing application in which M is a 703 by 592 sparse and rank deficient matrix and M^TM is full, the smaller problem consisted of a dense, rank deficient system comprising 231 equations and 120 variables. The application of our direct method to the smaller problem produced a solution to the original sparse linear least squares problem in a number of floating point computations tenfold smaller than a method that did not exploit the sparsity structure of M.

The algorithms derived from our decomposition method are well-suited for both sequential and parallel architecture machines. This work covers results obtained obtained from implementations on a sequential machine, whereas future work will cover results obtained from implementations on parallel architecture machines.

This report is organized as follows: Section 2 reviews some of the most common methods for solving the sparse linear least squares problem. These include the normal equations method, sequential row orthogonalization method (George & Heath, 1980) and multifrontal QR factorization (Liu, 1986). Section 3 presents the main decomposition result; Section 4 discusses various sparsity structures suitable for the proposed decomposition result including the block bordered triangular form; Section 5 gives two high-level implementations of the decomposition result; Section 6 reviews four methods available in the linear algebra package Matlab (Mathworks, 1990) for solving dense and sparse linear least squares problems; Section 7 presents a procedure to structure the overdetermined matrix arising from the signal processing application into the block bordered triangular form, and discusses the key computational issues in permutating an arbitrary nonsquare matrix into block bordered triangular form; Section 8 outlines some of the parallel features inherent in our algorithms, and concludes with numerical results and comparisons.

2. BACKGROUND AND MOTIVATION

The system of normal equations

$$M^T M x = M^T b \tag{3}$$

plays a pivotal role in many methods developed for the solution of the linear least squares problem. If M has full column rank, then the solution of the linear least squares problem (2) is given by the solution of the system of normal equations (3). This leads to the following three important contributions (George & Heath, 1980; Heath, 1984; Liu, 1986) for solving the sparse linear least squares problem.

1. Normal Equations Method. For any n by n permutation matrix P, $P^T TM^T MP$ is symmetric positive definite since matrix M has full column rank. As a result, the normal equations method leads to the following algorithm (Heath, 1984) for the solution of the sparse linear least squares problem.

```
procedure normal_equations:
begin
  determine symbolic structure of MTM;
  find permutation matrix P so that $P^T M^T MP$ has sparse upper Cholesky factor;
  factor $P^T M^T MP$ symbolically;
  comment this generates a row-oriented data structure for Cholesky factor;
  compute MTM and MTb numerically;
  perform Cholesky factorization $P^T M^T MP = R^T R$;
  solve $R^T z = P^T M^T b$, RY = z and x = Py in that order
end
```

Although the normal equations method is appealing for its simple formulation, it may give rise to serious loss of information in the explicit computation of the products $M^T M$ and $M^T b$. Also, the condition number of $M^T M$ is the square of the condition number of M, and so an accurate solution of the system (1) may be extremely difficult to compute if M is ill-conditioned. To avoid these potential numerical instabilities associated with the normal equations method, George and Heath (1980) combined orthogonalization with the normal equations method to arrive at the following important method for solving the sparse linear least squares problem.

2. Sequential Row Orthogonalization Method. Let Q be an m by m orthogonal matrix and let P be any n by n permutation matrix so that

$$Q^T MP = \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{4}$$

where R is an n by n upper triangular matrix and 0 is m-n by n zero matrix. Combining (1) and (4), and partitioning $Q^T b$ conformably into the direct sum of f and h, gives rise to the upper triangular system

$$Ry = f \tag{5}$$

in which $y = P^T x$. The vector x = Py obtained subsequent to the solution of (5) is a solution to the linear least squares problem (2) since the 2-norm is preserved under orthogonal transformations (Golub, 1965). The matrix Q is usually obtained using either Householder reflections or Givens rotations, or by Gram-Schmidt orthogonalization (Golub & Van loan, 1989; Lawson & Hanson, 1974).

While orthogonalization eliminates the potential numerical instabilities encountered in the normal equations method, the application of orthogonal transformations to the matrix MP may cause severe fill-in. George and Heath (1980), however, noted the following identity

$$P^T M^T M P = R^T R, \tag{6}$$

which clearly shows that the upper triangular matrix R obtained from the application of orthogonal transformations to the matrix MP is the upper Cholesky factor of the symmetric positive definite matrix $P^T M^T MP$. This observation led George and Heath to an algorithm which combines the attractive features in the normal equations method and orthogonalization into the following algorithm.

```
procedure sequential_row_orthogonalization:
begin
  apply the first three steps in procedure normal_equations
  compute R and f by applying Givens rotations to rows of [MP b] one at a time;
  solve Ry = f and x = Py in that order
end
```

The most significant contribution of the sequential row orthogonalization method is highlighted in step 2 of the above procedure where orthogonal transformations are carried out using fixed (static) data structure for R (George & Heath, 1980). This feature makes the sequential row orthogonalization method extremely efficient for sparse linear least squares problems when $M^T M$ is assumed to be sparse.

3. Multifrontal QR Factorization Method. This method, originally called row merging scheme (Liu, 1986), is a means to compute the upper triangular matrix R in the QR factorization (4) by applying Householder reflections to small dense submatrices in such a way that each submatrix is factorized independently of the others. The allocation of these small submatrices (called frontal matrices) and the subsequent treatment at the completion of their factorizations is accomplished by exploiting the sparsity structure of the symmetric matrix $R+R^T$.

   In what follows, we highlight the main motivation that has led to the formulation of the multifrontal QR factorization method. For any i with $1 \le i \le n$, let r_i and c_i be two arrays so that $M(r_i, c_i)$ is the submatrix of M consisting of the rows containing the nonzeros in the ith column of M, and the columns containing at least one nonzero in the submatrix $M(r_i,:)$. Then, the first row obtained from the QR factorization of the submatrix $M(r_i, c_i)$ is the first row of matrix R in (4) for the case where P = I. Now let M(:,j) be any other column of M such that

$$M(:,i)^T M(:,j) = 0. \tag{7}$$

   Assume that no cancellation of nonzero elements takes place in (7). Then no row of M contains nonzero entries in both M(:,i) and M(:, j), and so the first row obtained from the QR factorization of the submatrix $M(r_j, c_j)$ is also a row of the matrix R in (4) for the case where P = I. Consequently, for the case where P = I the first two rows of R can be obtained by independent QR factorizations of the submatrices $M(r_i, c_i)$ and $M(r_j, c_j)$. Such submatrices are called frontal matrices, and the method derived from the use of frontal matrices takes the following algorithmic form.

```
procedure multifrontal_QR_factorization:
begin
  apply the first two steps in procedure normal_equations;
  set M to MP;
  while M has any column do
     begin
        determine frontal matrices in M;
        for each frontal matrix M in M do
           begin
               use Householder reflections to compute QR factorization $Q^T M$;
               comment first row of $Q^T M$ is a row of R in (4);
               replace submatrix M of M by factorized frontal matrix $Q^T M$
           end;
        delete from M first row and column of each factorized frontal matrix
     end
end
```

By equality (7), each pass of the **for** loop in the above procedure can be done independently of the other passes. This feature makes the multifrontal QR factorization method well-suited for parallel computation. However, the overall effectiveness of the method rests on how efficiently the frontal matrices in M are allocated and formed at each pass of the **while** loop. This subject is covered next using a graph-theoretic setting. Our definitions are from Tarjan, 1983).

Let G = (V, E) be the directed graph of the upper Cholesky factor R. Then G is an acyclic graph since R is an upper triangular matrix, and so there exists in G at least one vertex v such that no edge in G enters v. A vertex such as v is called a root. Let X denote the set of roots in G. We will now show that each root in G leads to a distinct frontal matrix. These are the frontal matrices (|X| in total) that are determined at the first line of the **while** loop, and factorized at the first pass of the **for** loop.

Let G' = (V, E') be the undirected graph of the symmetric matrix R + R^T. We want to show that X is an independent set (no two vertices are adjacent) in G'. Assume for contradiction that there exists in X a pair of vertices v and w so that v is adjacent to w in G'. Then (v, w) is an edge in G', and so by the construction of G and G' it follows that there exists in G a directed edge that either leaves v and enters w or leaves w and enters v. In each case, we have established a contradiction since both v and w are roots in G. Consequently, X is an independent set in G', which means that X is an independent set in the undirected graph of $P^T M^T MP$ since this graph is a subgraph of G'. But if X is an independent set in the undirected graph of $P^T M^T MP$, then each pair of columns in MP corresponding to a pair of vertices in X satisfy relation (7). Hence, each root in G leads to a frontal matrix in MP.

Let M be the m – |X| by n – |X| matrix obtained at the completion of the last step in the **while** loop. By the property that the first row of each factorized frontal matrix in the first pass of the **for** loop is a row of R in (4), the second pass of the **for** loop will p roduce the next set of rows of R in (4). This process is repeated until all n columns of M have been deleted, in which case the **while** loop terminates and the computation of the upper triangular matrix R in (4) is completed.

In what follows, we cover some of the graph-theoretic details needed to form the frontal matrices required at the start of the second through last pass of the **for** loop, and also to connect our interpretation with the one that is usually used to describe the multifrontal QR factorization method.

Let us set X_0 to X, and define X_1 as the set of roots in the induced subgraph $G(V–X_0)$. Since no vertex in X_1 is in the initial set of roots X_0, no vertex in X_1 is a root in G, which means that for every root x in $G(V–X_0)$, there is at least one edge in G which leaves a root in G and enters x. We will call the vertex v in X_0 a parent of the vertex x in X_1. With this notation in hand, it is not hard to show that the frontal matrix associated with a root x in $G(V–X_0)$ is derived from the frontal matrices associated with the parents of x in X_0. It is important to note, however, that no two distinct vertices x and y in X_1 have a common parent in X_0. Otherwise, the pair (x, y) will be an edge in the induced subgraph $G(V–X_0)$ which contradicts the assumption that x and y are roots in $G(V–X_0)$.

We complete the description of the multifrontal QR factorization method with the following algorithm.

```
procedure roots:
begin
  compute the set of roots $X_0$ in G;
  V ← V – $X_0$;
  i ← 0;
  while V is not empty do
     begin
        i ← i + 1;
        compute the set of roots $X_i$ in G(V);
        for each each x in $X_i$ do compute the set of parents of x in $X_{i-1}$;
           V ← V – $X_i$
     end
end
```

Let h denote the value of the integer i at the completion of procedure roots. Then X_0, X_1, ..., X_h are the sets of roots computed at the completion of procedure roots. Note that all parents of a root in any X_i are in the set of roots X_{i-1} for all $i > 0$. This means that any edge that has its end points in two non-consecutive sets of roots is redundant as far as the computation of a frontal matrix is concerned. Let $G^* = (V, E^*)$ be the graph obtained from G by deleting all such edges. Then the undirected version of G^* obtained from G by replacing each directed edge by an undirected edge is the elimination tree of the undirected graph G', and h is the height of the elimination tree. An elimination tree is usually the means for describing the multifrontal QR factorization method (Liu, 1986). However, this approach requires several constructs which were not needed in our interpretation.

The problem of finding a permutation matrix P such that P^TM^TMP has sparse upper Cholesky factor is crucial in each of the three methods covered herein. Therefore, the assumption that M^TM is a sparse matrix is critically important since these methods may behave very poorly for the case where the matrix M^TM is dense.

In this work, we expand the choice of methods for solving the sparse linear least squares problem by developing a method that efficiently handles the case where M is a sparse matrix while M^TM is dense, and possibly full. The need to deal with this case was motivated by an important Navy signal processing application in which M is a highly sparse large overdetermined matrix and M^TM is completely full.

3. A DECOMPOSITION BASED ON THE SPARSITY OF M

Let P and S be permutation matrices such that PMS is a 2 by 2 block matrix

$$PMS = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{8}$$

in which the leading block A is an N by N square and nonsingular matrix. Replacing M by PMS in the original system of equations Mx = b, we obtain the following equivalent nonsquare system

$$(PMS)(S^T x) = (Pb) . \tag{9}$$

Consider this system where PMS has the 2 by 2 block form in (8) and the vectors $S^T x$ and Pb have been partitioned conformably into the direct sums of y and z, and u and v respectively. The system (9) then becomes

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} . \tag{10}$$

Now let K be the 2 by 1 block matrix defined by

$$K = \begin{bmatrix} A \\ C \end{bmatrix} . \tag{11}$$

Since block A is nonsingular, matrix K has full column rank and so there exists an orthogonal matrix Q such that $Q^T K$ has the following 2 by 1 form

$$Q^T K = \begin{bmatrix} R \\ 0 \end{bmatrix} , \tag{12}$$

in which R is an N by N upper triangular matrix with nonzero diagonal entries and 0 is a zero matrix. Define now the following identity

$$\begin{bmatrix} X & f \\ \Delta & h \end{bmatrix} = Q^T \begin{bmatrix} B & u \\ D & v \end{bmatrix} , \tag{13}$$

where X, Δ, f and h have the same dimensions as B, D, u and v respectively. Then, by (9) through (13), it follows that the following two nonsquare systems of equations

$$(Q^T PMS)(S^T x) = (Q^T Pb) \tag{14}$$

and

$$\begin{bmatrix} R & X \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} f \\ h \end{bmatrix} \tag{15}$$

are equivalent.

Since M is a nonsquare matrix and R is a square matrix, it follows that the block Δ in (15) is a nonsquare matrix, which means that the system of equations $\Delta z = h$ is nonsquare. With this property of block Δ, we are in position to state the main result. To facilitate the presentation of the result, we will designate the solution to problem (2) as a least squares solution of system (1).

Theoren 1. Suppose the m – N vector z* is a least squares solution of the nonsquare system

$$\Delta z = h .$$

Then the n vector x* defined by

$$x * = S\begin{bmatrix} R^{-1}(f - Xz *) \\ z * \end{bmatrix},$$

is a least squares solution of the nonsquare system of equations Mx = b.

Proof. Assume for contradiction that the n vector x* is not a least squares solution of the system Mx = b. Then there exists another n vector ζ such that

$$\| M\zeta - b \|_2 < \| Mx * - b \|_2 \ ,$$

and so we get

$$\| (Q^TPMS)(S^T\zeta) - (Q^TPb) \|_2 < \| (Q^TPMS)(S^Tx *) - (Q^TPb) \|_2 \ , \tag{16}$$

since the 2-norm is preserved under orthogonal transformations. Not let y* be the N vector such that

$$S^Tz * = \begin{bmatrix} y * \\ z * \end{bmatrix}. \tag{17}$$

Also, let us partition the n-vector $S^T\zeta$ conformably into the direct sum of Ψ and ζ. Then, by (14) through (17), we obtain the following inequality

$$(\| R\psi + X\zeta - f \|_2^2)^{1/2} < (\| Ry * + Xz * - f \|_2^2 + \| \Delta z * - h \|_2^2)^{1/2} . \tag{18}$$

By the assertion of the theorem and relation (17), however, we have

$$Ry * + Xz * - f = 0 . \tag{19}$$

So, by combining relations (18) and (19) we get the inequality

$$\| \Delta\zeta - h \|_2 < \| \Delta z * - h \|_2 \ ,$$

which is a contradiction since z* is a least squares solution of the nonsquare system Δz = h. This completes the proof.

Theorem 1 is applicable to both overdetermined and underdetermined systems of equations. Also, Theorem 1 is applicable to both full rank and rank deficient nonsquare matrices. If M has full rank, then the matrix Δ has full rank. Similarly, if M is rank deficient, then Δ is rank deficient, and so any linear least squares method chosen for the solution of the original nonsquare system of equations Mx = b can be used for the solution of the smaller nonsquare system Δz = h in Theorem 1.

In computer implementations of the decomposition in Theorem 1, we compute the orthogonal matrix Q in (12) as the product of two m by m orthogonal matrices Q_1 and Q_2 so that

$$Q^T = Q_2^TQ_1^T , \tag{20}$$

where Q_1 has the special form

$$Q_1^T = \begin{bmatrix} \mathbf{Q}^T & \\ & I \end{bmatrix} \tag{21}$$

in which I is an m – N by m – N identity matrix. This product form of Q allows us to use the N by N orthogonal matrix **Q** in Q_1 to exploit any special zero-nonzero structure that the leading block A might have. The other orthogonal matrix Q_2 in the product form is chosen so that

$$Q_2^T(Q_1^T K) = Q_2^T \begin{bmatrix} \mathbf{Q}^T A \\ C \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{22}$$

where R is an N by N upper triangular matrix with nonzero diagonal entries and 0 is a zero matrix.

The product form of Q in (20) is also suitable for dealing with the practical case where the leading block A in PMS is a singular matrix. This is done as follows. Let r denote the rank of the N by N singular block A. Without any loss of generality, assume that the leading r columns of A are the linearly independent columns. Then there exists an N by N orthogonal matrix **Q** such that

$$\mathbf{Q}^T A = \begin{bmatrix} R' & \alpha \\ 0 & 0 \end{bmatrix} \tag{23}$$

in which R' is an r by r upper triangular matrix with nonzero diagonal entires and the 0s are zero matrices. So, by combining (8), (21), and (23), we obtain the following 2 by 2 block matrix

$$Q_1^T PMS = \begin{bmatrix} R' & B' \\ C' & D' \end{bmatrix}. \tag{24}$$

The N – r by r zero block in the first column of $\mathbf{Q}^T A$ is a submatrix of the block C', while the N – r by N – r zero block in the second column of $\mathbf{Q}^T A$ is a submatrix of the block D'. For the case that M is an overdetermined matrix and the zero block in the first column of $\mathbf{Q}^T A$ occupies the last r rows of block C', the 2 by 2 block matrix in (24) takes the zero-nonzero structure shown in figure 1.

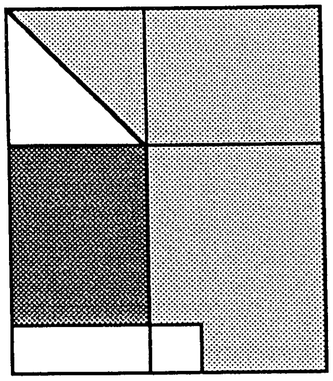

Figure 1. Zero-nonzero structure of $Q_1^T PMS$ when M is overdetermined.

Our next task is to zero the m – N by r dark shaded nonzero block below the upper triangular matrix in figure 1. Let K be the 2 by 1 block matrix defined by

$$K = \begin{bmatrix} R' \\ C' \end{bmatrix}. \tag{25}$$

Since R′ is an upper triangular matrix with nonzero diagonal entries, the matrix K has full column rank, and so there exists an orthogonal matrix Q_2 such that

$$Q_2^T K = \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{26}$$

in which R′ is an r by r upper triangular matrix with nonzero diagonal entries and 0 is m – r by r zero matrix. Define now

$$\mathbf{Q}_1^T Pb = \begin{bmatrix} R' \\ v' \end{bmatrix}, \tag{27}$$

where u′ and v′ are r and m - r vectors respectively. Also, let

$$\begin{bmatrix} X & f \\ \Delta & h \end{bmatrix} = Q_2^T \begin{bmatrix} B' & u' \\ D' & v' \end{bmatrix}, \tag{28}$$

where X, Δ, f and h have the same dimensions as B′, D′, u′ and v′ respectively. Then, by (20), (21) and (23) through (28), it follows that the nonsquare system of equations (15) is equivalent to the following nonsquare system

$$\begin{bmatrix} R & X \\ 0 & \Delta \end{bmatrix} \begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} f \\ h \end{bmatrix} \tag{29}$$

in which y′ and z′ are r and m – r vectors respectively. Figure 2 illustrates the zero-nonzero structure of the matrix $Q^T PMS$ when M is overdetermined.

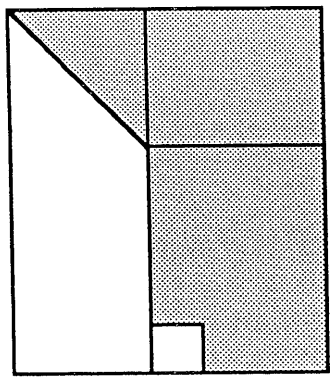

Figure 2. Zero-nonzero structure of $Q^T PMS$ when M is overdetermined.

Note that the N – r by N – r zero block in the block Di in figure 1 remains zero in the process of computing the block Δ in figure 2 since the N – r by r block right below the upper triangular matrix R′ in figure 1 is zero. Theorem 1 is now directly applicable to the case where the leading block A in PMS is a singular matrix.

4. EXPLOITING THE STRUCTURE OF LEADING BLOCK IN PMS

The zero-nonzero structure of the leading block A in the 2 by 2 block matrix PMS is central in the effective utilization of the decomposition in Theorem 1. If A is a dense matrix, then Theorem 1 provides no advantage over other methods for solving the sparse linear least squars problem. However, when A is a large sparse matrix with rich structure, the benefits gained from the application of Theorem 1 can be worthwhile.

Consider the most favourable situation where the leading block A in PMS is a block diagonal matrix. Then PMS is called a block bordered diagonal matrix (Zhang, Byrd & Schnabel, 1992). The following two observations highlight the key advantages in using a bordered block diagonal matrix to compute a least squares solution of the system Mx = b. First, each diagonal block of A can be orthogonalized independent of the remaining diagonal blocks. This means that the orthogonal matrix Q_1 in (20) can be computed so that each of the N diagonal blocks of A is handled by a different processor on a parallel architecture computer. Second, the matrix R in (12) and R' in (23) are block diagonal matrices, which means that the sparsity of the off-diagonal part of the block diagonal matrix A is fully preserved at the completion of orthogonalizations in (12) and (23).

It is important to note that each diagonal block of the block diagonal matrix A is a frontal matrix in M if the off-diagonal block C in PMS is a zero matrix. This is not hard to see since any two columns of A taken from two different diagonal blocks will satisfy equality (7) if C is a zero matrix. But if the diagonal blocks of A are frontal matrices, then by the graph-theoretic interpretation of frontal matrices given earlier there must exist in the undirected graph of the symmetric matrix M^TM an independent set of size N. However large independent sets are generally found in graphs that are quite sparse. Hence for the case where M^TM is assumed to be a dense or full matrix, the diagonal blocks of A may not be found by the conventional methods used for allocating frontal matrices.

The permutation of an arbitrary nonsquare matrix M into a block bordered diagonal form is a very difficult computational problem. Moreover, the existence of large block diagonal matrices in arbitrary nonsquare matrices have been seldom reported. Unlike nonsquare or nonsymmetric matrices, block bordered diagonal forms are abundant when the underlying sparse matrix is symmetric. Industrial applications in which block bordered diagonal forms are extensively utilized include domain decomposition, VLSI circuit design, structural engineering, and power system network problems (Zhang, Byrd & Schnabel, 1992). Also, given any sparse structurally symmetric matrix M, the work in Kevorkian (1993) gives a linear-time algorithm for computing permutation matrices P and S so that PMS has a block bordered diagonal form.

Another structural form of A that retains one of the attractive computational features of a block diagonal matrix is the block triangular form. In sharp contrast to block bordered diagonal matrices, 2 by 2 block matrices in which the leading block is a block triangular matrix are encountered in real industrial and government applications. One such application has been encountered in a Navy signal processing problem involving the prediction of bistatic target scattering from monostatic data (Schenk, 1968; Schenk, 1993; Schenk & Benthien, 1989). In this particular application, the Helmholtz integral equation together with given boundary conditions are discretized to arrive at the following algebraic system

$$Y = FX, \tag{30}$$

in which Y is a p by p complex symmetric matrix (scattering function) with known diagonal entries (monostatic data), F is a p by q complex matrix (far-field propogator function), and X is a q by p matrix (surface pressures). The objective is to use the monostatic data (diagonal of Y) and the principle of reciprocity (symmetry of Y) to estimate the surface pressures (matrix X). This is done next.

Suppose we equate the diagonal of Y with the diagonal of the p by q product matrix FX in (30). Then we get

$$F(i,:)X(:,i) = Y(i,i) \tag{31}$$

Next, let us equate the (i, j) off-diagonal element of the symmetric matrix FX with its (j, i) element. Then we have

$$F(i,:)X(:,j) = F(k,:)X(:,i) \quad i = 1,\ldots,p-1;\ j = i+1,\ldots,\ p \tag{32}$$

Equations (31) and (32) together form a system consisting of p(p+1)/2 linear equations and pq unknown variables. To put this linear system of equations in matrix form, we let x be the pq vector

$$x = \begin{bmatrix} X(:,1) \\ X(:,2) \\ \cdot \\ \cdot \\ X(:,p) \end{bmatrix}, \tag{33}$$

and let b be the pq vector

$$b = \begin{bmatrix} \mathrm{diag}(Y) \\ 0 \end{bmatrix}, \tag{34}$$

where 0 is a zero column vector. Then, by combining (31) through (34), we obtain the following nonsquare system of equations

$$Mx = b, \tag{35}$$

in which M is a p(p+1)/2 by pq complex matrix. For the special case where p = 37 and q = 16, M is a 703 by 592 complex matrix with the zero-nonzero structure shown in figure 3. This plot of matrix M was obtained using the “spy” utility in version 4 of the linear algebra package Matlab (Mathworks, 1990).

Three key properties characterize the sparse linear least squares problem in (35). These are as follows:

a. M^TM is a full matrix,

b. M contains q nonzeros in first p rows and 2q nonzeros in remaining rows,

c. M is a rank deficient matrix.

By property (a), all three methods of normal equations, sequential row orthogonalization and multifrontal QR factorization would produce poor results for this signal processing application. By property (b), there does not exist in M a small subset of rows whose deletion will render the matrix M^TM sparse. By property (c), the bistatic target scattering problem is an inherently difficult problem whose solution may require the use of singular value decomposition at some stage of the computation process. This feature will be covered in more detail later on.

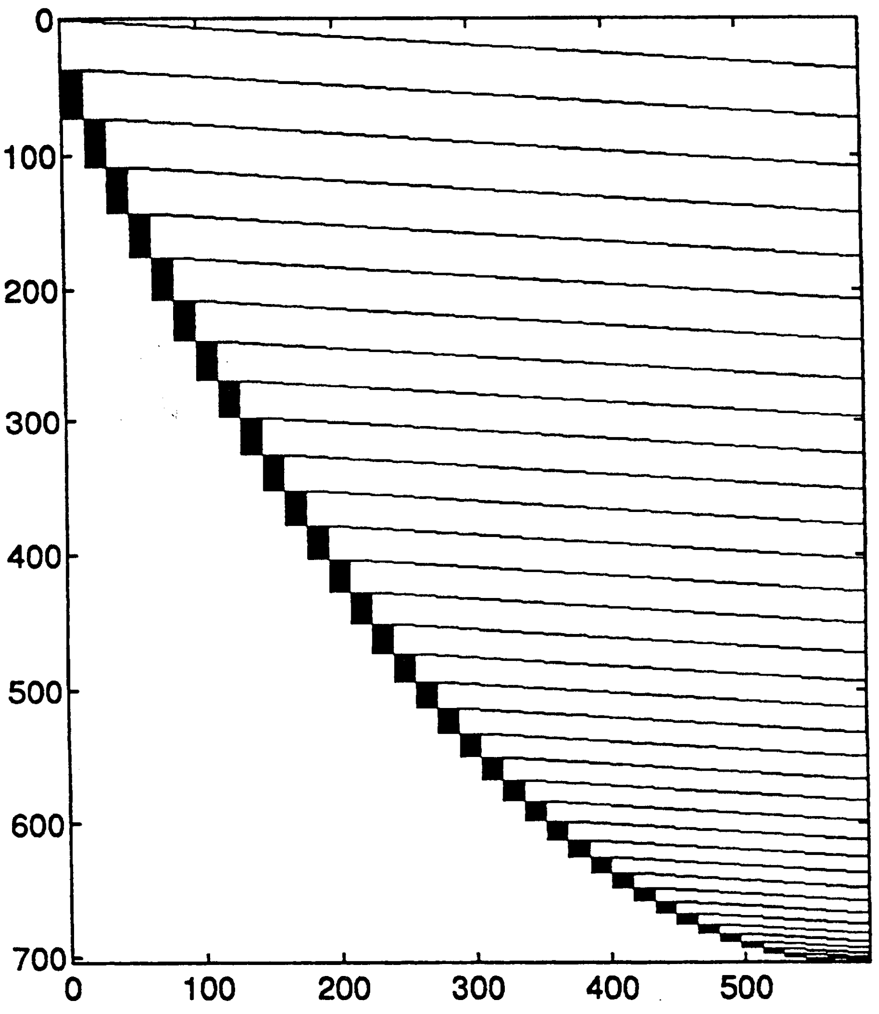

Figure 3. Unstructured form of bistatic target scattering problem.

Exploitation of the sparsity structure of matrix M in figure 3 has led us to a computer implementation that generates permutation matrices P and S so that the N by N leading block A in PMS is a p by p block upper triangular matrix with the following two distinct properties: (1) all p diagonal blocks are full square matrices, and (2) the leading p-q+1 diagonal blocks of A are q by q matrices while the remaining q-1 diagonal blocks have sizes q-1, q-2, ..., 2, 1 in that order. As an immediate consequence of the second property of the leading block A in PMS, we obtain

$$N = pq - q(q - 1)/2 \,. \tag{36}$$

Thus, for the case where p = 37 and q = 16, we have N = 472, which shows that a large portion of the original 703 by 592 matrix M is a block upper triangular matrix. Figure 4 shows the zero-nonzero structure of the 2 by 2 block matrix PMS. Consistent with the definition of a block bordered diagonal matrix, we call a 2 by 2 block matrix PMS a block bordered triangular matrix if the leading block A of PMS is block triangular.

Suppose PMS is any block bordered triangular matrix in which the leading block $A = [A_{ij}]$ is a p by q block upper triangular matrix with square diagonal blocks. Then, the following result highlights a numerical property of PMS that will be frequently used in subsequent developments.

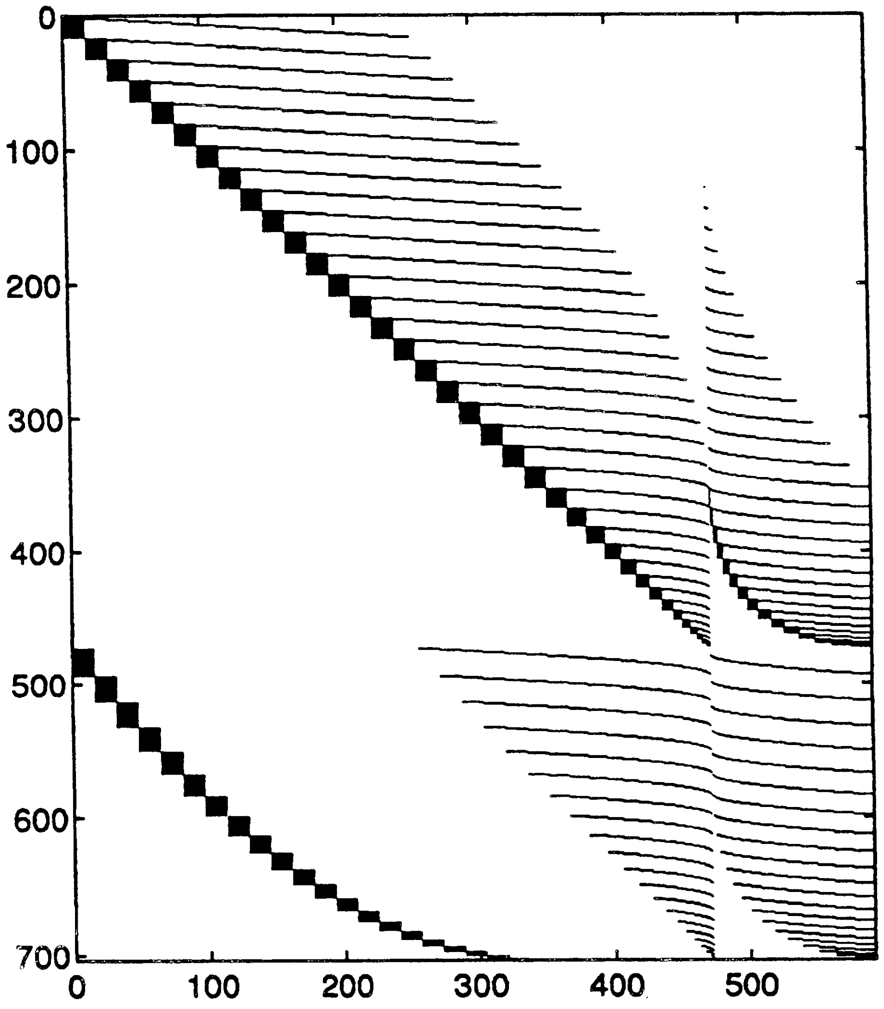

Figure 4. Structured form of bistatic target scattering problem.

Lemma 1. Let $\mathbf{Q}_i$ be an orthogonal matrix such that $\mathbf{Q}_i^T A_{ii}$ is upper triangular, and

$$Q = \begin{bmatrix} \mathbf{Q}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{Q}_p \end{bmatrix}.$$

Then the N by N matrix $\mathbf{Q}^T A$ is upper triangular.

Proof. The proof is obtained by premultiplying A by $\mathbf{Q}^T$.

5. IMPLEMENTATION OF THE DECOMPOSITION IN THEOREM 1

In this section, we use the decomposition in Theorem 1 to give detailed algorithms to solve the sparse linear least squares problem for the case where M^TM is dense. To make the algorithms directly applicable to the bistatic target scattering problem (35), we assume that the leading block A in the 2 by 2 block matrix PMS is a block triangular matrix with dense diagonal blocks. All algorithms will be presented in the Algol-like language in Aho, Hopcroft & Ullman (1976) adopted earlier.

Let k_i denote the order of the ith diagonal block of the p by p block triangular matrix $A = [A_{ij}]$ in PMS. The purpose of the orthogonal matrix Q_1 in (20) is to zero the strictly lower triangular parts of the p diagonal blocks of A. Thus, the premultiplication of A by the transpose of the orthogonal matrix Q_1 will zero all but top element of the $k_i - j + 1$ vector x defined by

$$x = A_{ii}(j : k_i, j) \quad i = 1, ..., p; j = 1, ..., k_i - 1 . \tag{37}$$

The other orthogonal matrix Q_2 in the product (20) is called upon to zero block C_i in the 2 by 1 block matrix K in (25), and so the premultiplication of K by the transpose of Q2 will zero all but top element of the m – N + 1 vector x defined by

$$x = \begin{bmatrix} R'(j, j) \\ C'(:, j) \end{bmatrix} \qquad j = 1, ..., N . \tag{36}$$

The vector x in both (37) and (38) is a dense subcolumn in the block bordered triangular matrix PMS.

Since M^TM is a dense or full matrix, the matrix R in (15) and (29) is a full upper triangular matrix, and so there are two ways to proceed with the implementation of Theorem 1. First, set up a row-oriented data structure for a full upper triangular matrix R, and subsequently use the sequential row orthogonalization method in (George & Heath, 1980) to compute R by applying Givens rotations to rows of [PMS Pb] one at a time. Second, avoid the use of sparse data structure since all their submatrices R, X and Δ in (15) and (29) are full matrices. In this work, we choose the second approach for the implementation of Theorem 1 since memory savings brought about from the use of sparse data structure is not substantial for problems where M^TM is a dense or full matrix.

Let RS and CS be two single arrays so that

$$M(RS, CS) = [PMSPb] .$$

Also, let IA be a (p+1) array defined by

$$IA = [1, 1 + k_1, ..., 1 + \textstyle\sum_1^p k_i] ,$$

in which the leading p elements are the pointers to the 1st row of the p diagonal blocks of the block triangular matrix A whereas the (p+1)th element is the pointer to the first row of blocks C and D in PMS as well as the first column of blocks B and D. From these three relations, one can readily allocate all diagonal blocks of A, blocks B, C, and D and the sub-vectors u and v of Pb as given in (10).

For clarity of presentation, we first give an implementation of Theorem 1 in which we assume that the leading block A is nonsingular. Next, we cover the more challenging case where the leading block A is assumed to be singular.

5.1 LEADING BLOCK A IS NONSINGULAR

The following algorithm implicitly computes the two orthogonal matrices Q_1 and Q_2 defined in relation (20). The integers a and z in procedure q_en_q designate the first and last rows of each diagonal block A considered in the algorithm.

```
procedure q_en_q:
begin
  comment compute Q (and so $Q_1$);
  for i ← 1 until p do
    begin
      a ← IA(i)
      z ← IA(i+1) – 1;
      for j ← a until z – 1 do
        hshldr(RS(j:z))
        end;
    comment compute $Q_2$;
    for j ← 1 until N do
      hshldr([RS(j), RS(N+find(M(RS(N+1: m), CS(j)) ≠ 0))])
end
```

The procedure hshldr(rows) called in q_en_q uses Householder reflections to zero all but the top element of the column vector w = M(rows, CS(j)). If rows = RS(j: z), then x is the vector given in (37). In the same way, if rows = [RS(j), RS(N+find(M(RS(N+1: m), CS(j)) ≠ 0))], then x designates the nonzero part of the vector given in (38). The function “find” is a Matlab utility for finding the indices of the nonzero elements in a vector.

```
procedure hshldr(rows):
begin
  x ← M(rows, CS(j));
  norm ← $||x||_2$;
  if normx > 0 then
    begin
      δ ← x(1) + sign(x(1))*normx;
      v ← [1; x(2:|x|)/δ];
      w ← –(2/($v^T$*v))*$v^T$*M(rows, CS(j:n+1))
      M(rows, CS(j:n+1)) ← M(rows, CS(j:n+1)) + v*w
    end;
end
```

Given x = M(rows, j), procedure hshldr(rows) computes a Householder vector v so that the first element of the vector $(1 = 2vv^T/v^Tv)x$ is nonzero and all other elements are zero. The Householder vector v computed in hshldr(rows) is defined in the standard way (Lawson & Hanson, 1974) as

$$v = x + \text{sign}(x(1)) \, \| x \|_2 \, e_1$$

where e_1 is the first column of |x| by |x| identity matrix. The signum function “sign” ensures that $| v(1) | = | x(1) | + \| x \|_2$, which means that $\| v \|_2 \geq \| x \|_2$, and so large relative errors in the coefficient $2/v^Tv$ can be avoided in the process of computing the product $(1 – 2vv^T/v^Tv)x$. In addition to this standard practice, we also follow a guideline established in Golub & Van Loan, 1989) to normalize the vector v so that v(1) = 1.

Since x = M(rows, CS(j)), the submatrix M(rows, CS(j:n+1)) can be written in the following augmented form

$$M(\text{rows},\ CS(j : n + 1)) = [x \ \ M(\text{rows},\ CS(j + 1 : n + 1))] \, .$$

Combining this with the last two lines in procedure hshldr we obtain the equality

$$M(\text{rows},\ CS(j : n + 1)) = (1 - \frac{2vv^T}{v^Tv})[x\ M(\text{rows},\ CS(j + 1 : n + 1))] .$$

Thus, at the completion of procedure hshldr, the vector $(1 – 2vv^T/v^Tv)x$ forms the first column of the submatrix M(rows, CS(j:n+1)). Therefore, if x is the vector in (37), then all the elements below the main diagonal in the jth column of block A_{ii} will be zero at the completion of procedure hshldr(rows). Similarly, if x is the vector given in (38), then all the elements below the main diagonal in the jth column of matrix PMS will be zero. Hence at the completion of procedure q_en_q we will obtain the following three identities.

$$\begin{aligned} R &= M(RS(1 : N), CS(1 : N)) , \\ [X\ f] &= M(RS(1 : N), CS(N + 1 : n + 1)) , \\ [\Delta\ h] &= M(RS(N + 1 : m), CS(N + 1 : n + 1)) . \end{aligned}$$

At this point, we are in position to compute a least squares solution of the smaller nonsquare system $\Delta z = h$ in Theorem 1. However, before we do this, we require the following result pertaining to the upper triangular matrix R.

Lemma 2. Suppose the N by N leading block A in the block bordered triangular matrix PMS is nonsingular. Then the upper triangular matrix R computed in procedure q_en_q has nonzero main diagonal.

Proof. Since the block A has ful lrank, the N by N leading block $\mathbf{Q}^TA$ is an upper triangular matrix R' with a nonzero main diagonal at the completion of the first **for** loop in procedure q_en_q. Consider now any call to hshldr(rows) in the second **for** loop in procedure q_en_q. Then x = M(rows, CS(j)) consists of the vector given in (38). Thus $R'(j, j)$ is the element at the top of vector x which is nonzero since R' has a nonzero main diagonal. This means that the top element in the vector $(1 – 2vv^T/v^Tv)x$ is also nonzero. But by the construction of the vector x, R(j, j) is the element at the top of the vector $(1 – 2vv^T/v^Tv)x$. Hence, the upper triangular matrix R has a nonzero main diagonal at the completion of the second **for** loop in procedure q_en_q. This completes the proof.

By Lemma 2, the condition normx > 0 is satisfied at each call to hshldr, and so for the case where the leading block A is nonsingular, the line “**if** normx > 0 **then**” in procedure houshldr can be deleted.

5.2 COMPUTING A LEAST SQUARES SOLUTION OF ΔZ = H

Choosing a method for computing a least squares solution of the sparse nonsquare system Mx = b strongly depends on the rank of the matrix M. If M has full rank, then the block Δ has full rank, and so one can obtain a least squares solution of the nonsquare system $\Delta z = h$ either by QR factorization or by Cholesky factorization of the system of normal equations

$$(\Delta^T\Delta)z = (\Delta^Th) . \qquad (39)$$

Thus, if z* is the solution of the triangular system resulting from the QR factorization or the solution of the normal equation (39), then the n vector x* defined in Theorem 1 is a least squares solution of the original nonsquare system of equations Mx = b.

In the bistatic target scattering application, the overdeterined matrix M is rank deficient and so the block Δ is rank deficient too. Therefore, the solution of the system of equations $\Delta z = h$ by QR factorization will break down since a diagonal element in the triangular system will be zero. Similarly, the use of the system of normal equation (39) will fail since the matrix $\Delta^T\Delta$ is singular which means that the Cholesky factorization will break down.

In sharp contrast to the QR factorization, normal equations and other linear least squares methods, the singular value decomposition (SVD) is applicable to full rank as well as rank deficient matrices. In view of this practical consideration, the SVD is our choice for the solution of the nonsquare system $\Delta z = h$.

Given any m by n nonsquare matrix M with rank r, there exists an m by m orthogonal matrix U and an n by n orthogonal matrix V so that $U^T MV$ is a diagonal matrix Σ with r positive diagonal elements in decreasing order and n – r zeros (Forsyth & Moler, 1967). Thus, if σ_1 through σ_n denote the diagonal elements of Σ, then we have

$$\sigma_1 \geq \sigma_2 ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0 .$$

The numbers σ_1 through σ_n are called the singular values of M.

Now let U and V be orthogonal matrices so that $U^T DV$ is a diagonal matrix Σ. Also, let r denote the rank of Δ. Then the nonsquare system of equations $\Delta z = h$ can be written in the following equivalent form

$$(U \sum V^T)z = h . \tag{40}$$

Now since the block Δ is a matrix with rank r, the leading r diagonal elements of are nonzero whereas the remaining n – r elements are zero. This means that the system of equations (40) can be written as

$$U(:, 1 : r) \sum(1 : r, 1 : r)V^T(1 : r, :)z = h . \tag{41}$$

Thus, if z* denotes the solution of this system of equations, then we obtain

$$z * = (:, 1 : r) \sum(1 : r, 1 : r)^{-1}U(:, 1 : r)^T h . \tag{42}$$

The vector z* is a least squares solution of the nonsquare system $\Delta z = h$.

With these results, the implementation of the decomposition in Theorem 1 takes the following algorithmic form when the leading block A in PMS is nonsingular.

```
procedure ls_dec:
begin
  use SVD to compute $\Delta = U * \sum * V^T$;
  r ← rank(Σ);
  z* ← $V(:, 1 : r) * \sum(1 : r, 1 : r)^{-1} * U(:, 1 : r)^T * h$;
  y* ← $R^{-1} * (f - X * z *)$;
  x* ← $S * [y *; z *]$;
end
```

5.3 LEADING BLOCK A IS SINGULAR

For the case in which the block upper triangular matrix A is singular, our primary objective is to use the orthogonal matrix Q_1 to detect the columns that are linearly independent in each of the diagonal blocks of A. This information is then used to update arrays RS and CS so that the premultiplication of matrix M(RS, CS) by the transpose of the orthogonal matrix Q_1 gives rise to the 2 by 2 block matrix given in (24) in which the leading block R' is an upper triangular matrix with nonzero diagonal entries.

For the detection of linearly independent columns, we use a Boolean array VC (Visited Columns) setting VC(j) = 1 if and only if the column of block A is a linearly independent column in a diagonal block of A. The entire algorithm called uls_dec (Unrestricted Least Squares Decomposition) is presented below.

```
**procedure** uls_dec:
**begin**
  VC ← zeros(1, N);
  q_and_q;
  use SVD to compute $\Delta = U * \sum * V^T$;
  r ← rank(Σ);
  z* ← $V(:, 1 : r) * \sum(1 : r, 1 : r)^{-1} * U(:, 1 : r)^T * h$;
  y* ← $R^{-1} * (f - X * z *)$;
  x* ← $S * [y *; z *]$;
**end**
```

Comparison of procedures ls_dec and uls_dec shows that the key difference between the two procedures is the replacement of procedure q_en_q by q_and_q and the introduction of array VC. The procedure q_and_q is given below.

```
**procedure** q_and_q:
**begin**
  **comment** compute **Q** (and so $Q_1$)
  **for** i ← 1 **until** p **do**
    **begin**
      a ← IA(i);
      z ← IA(i+1) - 1
      root ← a;
      **for** j ← a **until** z do
          hshldra(RS(root: z))
    **end** ;
  **comment** update arrays RS and CS;
  LI ← find(VC == 1);
  NN ← |LI|;
  **if** NN < N **then**
    **begin**
      LD ← find(VC == 0);
      RS ← [RS(LI), RS(N+1: m), RS(LS)];
      CS(1:N) ← [CS(LI), CS(LD)];
      N ← NN
    **end**
  **comment** compute $Q_2$;
  **for** j ← 1 **until** N do
      hshldrc([RS(j), RS(N+find(M(RS(N+1: m), CS(j))  0))])
**end**
```

There are three distinct parts in procedure q_and_q (highlighted by the **comment** lines). The first and third parts (which compute the orthogonal matrices Q_1 and Q_2 respectively) are derived from procedure q_en_q, whereas the second part in procedure q_and_q concerns the update of the arrays

CS and RS using array VC. The computation of the array VC is carried out in procedure hshldra called in the first part of q_and_q. This procedure is given below.

```
procedure hshldra(rows):
begin
  x ← M(rows, CS(j));
  normx ← ||x ||$_2$;
  if normx > 0 then
     begin
        VC(j) ← 1;
        root ← root + 1;
        $\sigma$ ← x(1) + sign(x(1))*normx;;
        v ← [1; x(2:| x |)/$\sigma$];
        w ← -(2/(v$^T$*v))*v$^T$*M(rows, CS(j:n+1));
        M(rows, CS(j:n+1)) ← M(rows, CS(j:n+1) + v*w
  if NN < N then
     begin
        LD ← find(VC == 0);
        RS ← [RS(LI), RS(N+1: m), RS(LS)];
        CS(1:N) ← [CS(LI), CS(LD)];
     end
end
```

Each time the condition normx > 0 is satisfied, the vector x is nonzero, which means that the jth column of the ith diagonal block of the block upper triangular matrix A is linearly independent. Consequently, VC(j) is set to 1 and the integer root is incremented by 1. The purpose of the integer root is to insure that the triangularization of each diagonal block of A is properly done. Note that procedures hshlkdra and hshldr are the same if the first two lines in the **begin** block of procedure hshldra are deleted.

At the completion of the orthogonal matrix Q_1, the computation of the array VC in procedure q_and_q is complete. Subsequently procedure q_and_q begins with its next task which is to update arrays RS and CS so that the matrix Q_1M(RS,CS) has the 2 by 2 block form given in (24). To do the update, we construct an array ILIC of pointers to the linearly independent columns detected in procedure hshldra, and let NN = | LI |. Thus, if NN = N, then we conclude that the leading block A in PMS is nonsingular, and so we skip the part in procedure q_and_q that involves the update of the arrays RS and CS.

Suppose NN < N. Then, the leading block A in PMS is singular, and so we proceed with the update of the arrays RS and CS in procedure q_and_q. To begin with, we construct an array LD of pointers to the columns of A that have not been marked as linearly independent. Now, by the construction of procedure hshldra, the NN by NN submatrix M(RS(LI), CS(LI) in M is an upper triangular matrix with nonzero diagonal entries at the completion of the first part in procedure q_and_q. Thus, to obtain the 2 by 2 block form in (24), we update arrays RS and CS so that rows RS(LI) and columns CS(LI) are moved to the front of arrays RS and CS respectively. As for the remaining rows and columns of block A, these are moved to the rear of array RS and sub-array CS(1:N) respectively so that the matrix M(RS, CS) has the 2 by 2 block form shown in figure 1 at the completion of the update. It should be noted however that the blank parts (zero submatrices) in blocks C' and D' shown in figure 1 may contain nonzero entries in this case since the triangularization in procedure hshldra is confined to the diagonal blocks of the block triangular matrix A. The advantages for executing hshldra in this way will be apparent when we discuss parallel implementations of the decomposition in Theorem 1 later on.

The purpose of the orthogonal matrix Q2 computed in the third part of procedure q_and_q is to zero the block C' given in (24). Procedure q_and_q accomplishes this task by calling procedure hshldrc given below.

```
procedure hshldrc(rows):
begin
  x ← M(rows, CS(j));
  normx ← ||x ||2;
  σ ← x(1) + sign(x(1))*normx;
  v ← [1;x(2:\ x |)/σ];
  w ← -(2/(vT*v))*vT*M(rows, CS(j:n+1));
  M(rows, CS(j:n+1)) ← M(rows, CS(j:n+1) + v*w
end
```

6. SOLVING THE LINEAR LEAST SQUARES PROBLEM USING MATLAB

The Matlab linear algebra package (Mathworks, 1990) provides an excellent computational platform for numerical comparisons. There are at least four distinct methods to compute a least squares solution of a nonsquare system of equations $Mx = b$ in Matlab. These are the following:

1. Normal Equations Method. The sparse Matlab implementation of the normal equations method is identical to the algorithm covered earlier in section 2. For the bistatic target scattering problem, the m by n matrix M is rank deficient, which means that the n by n matrix M^TM is singular and so the Cholesky factorization will break down. Also, M^TM is a full matrix and so the use of sparsity data structure for solving the bistatic target scattering problem by the normal equations method will be disasterous since a sparse data structure will be used to handle a large full matrix.
2. Augmentation Method. If we split the system of normal equations (3) into the following two systems

   $$\begin{aligned} r + Mx &= b \\ M^Tr &= 0 \end{aligned},$$

   then we arrive at the following 2 by 2 block matrix system

   $$\begin{bmatrix} I & M \\ M^T & o \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \qquad (43)$$

   proposed in Hachtel (1974) for solving the sparse linear least squares problem. If the identity matrix I and the vector r in (43) are replaced by αI and $(\alpha^{-1}r)$ respectively, where α is some nonzero constant, then we have the form used in Bjorck (1967) for computing linear least squares solution iteratively. In Matlab, the 2 by 2 block matrix in (43) is ordered by the minimum degree algorithm and then the system is solved by elimination. The augmented method is a complicated way of forming the system of normal equations, and so this algorithm will break down in the process of solving the bistatic target scattering problem since M is a rank deficient matrix.
3. QR Factorization. Since the overdetermined matrix M in the bistatic target scattering problem is rank deficient, the upper triangular matrix produced by the QR factorization of matrix M will have a zero on the main diagonal, and so this method will also break down in solving the upper triangular system.

4. Singular Value Decomposition. The singular value decomposition has been the only means in Matlab to produce reliable least squares solutions of the overdetermined system of equations in the bistatic target scattering application. For this important practical consideration, we have opted to avoid sparse data structures here to have a meaningful comparison basis for algorithms ls_dec, uls_dec, and method (IV) in Matlab.

7. COMPUTING PERMUTATION MATRICES P AND S

To promote the use of the bistatic target scattering overdetermined system of equations (35) for comparison purposes, we present a procedure called simsys(p, q) that simulates the overdetermined system (35) for any positive integers p and q. This simulated version of the bistatic target scattering problem retains all structural and numerical properties of the original problem. This procedure is as follows.

```
procedure simsys(p, q):
begin
  comment generate m by n zero matrix;
  m ← p*(p+1)/2;
  n ← p*q;
  M ← zeros(m, n);
  comment compute first p rows of 0–1 matrix M;
  for i ← 1 until p do
     M(i, 1+ (i – 1)*q: i*q) ← ones(1, q);
  comment compute remaining rows of 0–1 matrix M;
  for i ← 1 until p do
     begin
        α ← (i–1)*(2*p–i)/2;
        β ← 1+(i–2)*q:(i–1)*q;
        for j ← 1 until p do
           begin
              M(j + α, α) ← ones(1, q);
              M(j, 1+(j – 1)*q: j*q)← ones(1, q);
           end
     end
  comment compute single arrays RS and CS;
  p_and_s;
  comment select nonzero entries of matrix M and vector b;
  recast
end
```

For the case p = 37 and q = 16, the procedure simsys (excluding procedures p_and_s and recast) generates the 703 by 592 overdetermined matrix M shown in figure 3.

The procedure p_and_s called in simsys computes row and column sequences RS and CS so that

$$M(RS, CS) = [PMSPb] .$$

Procedure p_and_s also computes the single array IA used in procedures q_en_q and q_and_q for allocating the blocks in PMS and the subvectors in Pb. The entire procedure is given below.

```
procedure p_and_s:
begin
  comment initialize arrays RS, CS and IS and markers VR, VC;
  RS ← zeros(1, m);
  CS ← 1:n+1;
  IA ← [];
  VR ← zeros(1, m);
  VC ← zeros(1, n+1);
  comment compute parts of RS and CS needed to construct A;
  τ ← p;
  δ ← 1;
  k ← 0;
  a ← 1;
  for i ← 1 until p do
     begin
        IA(i) ← a;
        δ ← δ + τ;
        p ← min(τ, q)–1;
        z ← a + p;
        RS(a:z) ← [i, δ:δ + p – 1];
        if τ < q then CS(a:z) ← k+1 : k+τ;
        k ← i*q;
        τ ← τ–1;
        a ← z+1;
     end
  comment compute remaining parts of RS and CS;
     IA(p+1) ← a;
     N ← z;
     VR(RS(1:N)) ← ones(1:N);
     VC(RS(1:N)) ← ones(1:N);
     RS(N+1:m) ← find(VR==0);
     CS(N+1:n+1) ← find(VC==0);
end
```

For the case p = 37 and q = 16, the application of procedure p_and_s to the 703 by 592 overdetermined matrix in figure 3 produces the block bordered triangular matrix shown in figure 4.

The procedure recast called the completion of procedure p_and_s in simsys accomplishes four distinct tasks. These are as follows:

a. Fixing rank of leading block A. For any application of ls_dec, procedure recast modifies the main diagonal of A so that rank(A) = N. Procedure recast does this as follows. Since $m = p(p-1)/2$, $n = pq$ and $m > n$, we have

$$p > 2q + 1 . \tag{44}$$

Also, by the construction of the 0–1 matrix M in simsys, each row in the first p rows of M has exactly q 1s while each of the remaining m–p rows has exactly 2q 1s, and so by (44) we obtain

$$p > \sum_{i \neq j} |M(i, j)| , \qquad i = 1, ..., m .$$

Thus, if the main diagonal of A is modified so that each diagonal entry is set to p, then A becomes a positive definite matrix with rank(A) = N. Procedure recast accomplishes this task by using the relation

$$M(RS(i), CS(i)) = p\,, \qquad\qquad i = 1, ..., N\,,$$

since A = M(RS(1:N),CS(1:N)). For applications requiring a singular leading block in PMS, we modify A so that each of the p–q—1 diagonal blocks of A has exactly one linearly dependent column. All remaining q–1 diagonal blocks of A are converted to positive definite matrices by setting all diagonal elements of these diagonal blocks to p.

b. Modifying block D. We compute in the bipartite graph of block D a maximal matching, and subsequently we set the locations in M corresponding to the elements of the maximal matching to p. This task creates in D a positive definite matrix of size equal to the computed maximal matching.

c. Converting M to a complete matrix. In the original bistatic target scattering application, the matrix is complex and, so to create a similar environment in our numerical tests, we convert M to a complex matrix using the relation

$$M = M + 0.2 * (\sqrt{-1}) * M\,.$$

d. Choosing right-hand side vector b. We choose b so that component i of vector b is equal to the sum of the absolute values of the nonzero entries in row i of matrix M. For the case where A is nonsingular (applications of procedure ls_dec), this task combined with task (b) in procedure recast produces a least squares solution x in which each component is equal to 1.

7.1 COMPUTING P AND S FOR THE GENERAL CASE

Given an arbitrary nonsquare sparse matrix M, the finding of permutation matrices P and S such that PMS is block bordered triangular matrix is a computationally very difficult problem especially when certain constraints are imposed on the size of the leading block A in PMS. In the special case where M is a square matrix, the permutation of M to block bordered triangular form is closely related to the NP-complete problem of finding a minimum feedback vertex set in the directed graph of M.

In this section we give a simple greedy-type linear algorithm to put a nonsquare sparse matrix M into the block bordered triangular form PMS. The method proceeds as follows. First, compute in the bipartite graph of matrix M a maximal matching H. Let k = |H| and assume without any loss of generality that k < min(m,n). Then the maximal matching H gives rise to a permutation matrix P' so that $P'M$ is a 2 by 2 block matrix in which the leading block is a k by k matrix with nonzero main diagonal. Let G = (V, E) be the directed graph of the leading block in $P'M$. Second, use depth-first seach (Aho, Hopcroft & Ullman, 1976) to compute the strongly connected components of G in topologically sorted order. Let p be the number of strongly connected components in G. Then at the completion of the second step we have permutation matrices P'' and S such that the matrix $P''(P'M)S$ is a 2 by 2 block matrix in which the leading block is a p by p block triangular matrix of size k. This completes the construction of the block bordered triangular matrix PMS, where $P = P''P'$.

The parameters k and p dictate the usefulness of the block bordered triangular matrix PMS in Theorem 1. If the maximal matching H computed in the first step has a small cardinality, then the block triangular part in PMS will be small, and the advantages of the decomposition in Theorem 1 will be limited. Thus, to make the greedy algorithm more effective, one may require algorithms that produce large maximal matchings. Similarly, if G is a strongly connected graph or with few strongly connected components, then the leading block A in PMS will not have an interesting structure for the decomposition in Theorem 1 since the parameter p will be small.

For the case where G contains few strongly connected components, we suggest the following step. In the depth-first search tree of the directed graph G, there are three types of edges (Aho, Hopcroft & Ullman, 1976). These are forward edges, back edges, and cross edges. By construction, each back edge in the depth-first search tree gives rise to a cycle in G, and so the vertex incident with the largest number of back edges is common to a large number of cycles in G. Thus, if we remove such vertices from G, then the resulting graph may have much greater number of strongly connected components. This means that the leading block triangular matrix A in PMS may have a large number of diagonal blocks. This approach for increasing the size of the parameter p was an essential step into the block bordered triangular form shown in figure 4.

For a more rigorous treatment of this problem, more research effort is needed. We hope that the successful application of Theorem 1 to a real practical problem may be the stimulus for further research in this important area.

8. NUMERICAL RESULTS AND COMPARISONS

The linear least squares algorithms ls_dec and uls_dec are well-suited for parallel architecture machines. For example, the computation of the orthogonal matrix Q_1 in procedurs q_en_q and q_and_q can be readily done on a parallel machine as each of the p passes of the **for** loop (used for computing Q_1) is independent of the other p-1 passes. Also, the computation of the orthogonal matrix Q_2 can be done in such a way that the parallel architecture of a machine is fully explored. The parallel features of algorithms ls_dec and uls_dec will be the topic of further work in this area. Here, we present numerical results and comparisons to demonstrate the effectiveness and accuracy of these two algorithms on conventional machines.

Tables 1 and 2 summarize the results obtained from the application of algorithms ls_dec, uls_dec and the singular value decomposition in Matlab to five instances of the bistatic target scattering problem. Column 1 in these tables gives the parameters p and q used in procedure simsys to generate the m by n overdetermined matrix M. Column 2 includes the size of matrix M in each of the five examples while column 3 gives the size N of the nonsingular leading block A in the structured matrix PMS. Columns 4 and 5 give the number of floating-point operations (flops) required to compute a least squares solution while columns 6 and 7 give a measure of the accuracy of the computed results.

The results in tables 1 and 2 reflect the effectiveness of the decomposition result in Theorem 1 to compute a least squares solution of an overdetermined rank deficient system using singular value decomposition. The improvements on a parallel architecture machine are expected to be more substantial.

Table 1. Applications of ls_dec and Matlab to the bistatic target scattering problem.

(p, q)	m by n	N	flops (10^6)		$\|\|Mx - b\|\|_2$ (10^{-12})	
			ls_dec	Matlab	ls_dec	Matlab
(26, 10)	351 by 240	215	73.86	700.96	2.44	13.14
(30, 13)	465 by 390	312	185.87	2057.60	5.67	18.62
(37,16)	703 by 592	472	630.98	6764.70	13.36	21.57
(42, 20)	903 by 840	650	1413.20	16958.00	21.75	61.56
(44, 22)	990 by 968	737	1897.10	23350.00	27.12	62.74

Table 2. Applications of uls_dec and Matlab to the bistatic target scattering problem.

(p, q)	m by n	N	flops (10^6)		$\|\|Mx - b\|\|_2$ (10^{-12})	
			ls_dec	Matlab	ls_dec	Matlab
(26, 10)	351 by 240	198	92.04	731.99	3.24	4.97
(30, 13)	465 by 390	294	220.68	1995.10	8.29	7.86
(37,16)	703 by 592	450	720.31	6565.40	19.83	18.48
(42, 20)	903 by 840	627	1578.60	16663.00	37.35	22.46
(44, 22)	990 by 968	714	2109.70	*	45.53	*

* memory limitations

9. REFERENCES

Aho, A. V., J. E. Hopcroft, and J. D. Ullman. 1976. *The Design and Analysis of Computer Algorithms,* 3rd printing. Addison–Wesley, Reading, MA.

Bjorck, A. 1967. “Iterative refinement of linear least squares solutions,” *BIT,* 7, pp. 257–278.

Forsyth, G., C. B. Moler. 1967. *Computer Solution of Linear Algebraic Systems,* Prentice–Hall, Englewood Cliffs, NJ.

George, A., M. T. Heath. 1980. “Solution of sparse linear least squares problems using Givens rotations,” *Linear Algebra Appl.,* 34, pp. 69–83.

Golub, G. H. 1965. “Numerical methods for solving linear least squares problems,” *Numer. Math.*, 7, pp. 206–216.

Golub, G. H., C. F. Van Loan. 1989. *Matrix Computations*, The John Hopkins University Press, Baltimore.

Hachtel, G. D. 1974. “Extended applications of the sparse tableau approach—finite elements and least squares,” Technical Report, Elec. Sci. and Eng. Dept. and Comput. Sci. Dept., UCLA.

Heath, M. T. 1982. “Some extensions of an algorithm for Sparse linear least squares problems,” *SIAM J. Sci. Stat. Comput.*, 3, pp. 223–237.

Heath, M. T. 1984. “Numerical methods for large sparse linear least squares problems,” *SIAM J. Sci. Stat. Comput.*, 5, pp. 497–513.

Kevorkian, A. K. 1993. “Decomposition of large sparse symmetric systems for parallel computation. Part 1. Theoretical Foundations,” NCCOSC/NRaD Technical Report 1572.

Lawson, C. L., R. J. Hanson. 1974. *Solving Least Squares Problems,* Prentice–Hall, Englewood Cliffs, NJ.

Liu, J. H. 1986. “On general row merging schemes for sparse Givens transformations,” *SIAM J. Sci. Stat. Comput.*, 7, pp. 1190–1211.

The Mathworks. 1990. *Pro-Matlab User’s Guide,* South Natick, MA.

Peters, G., J. H. Wilkinson. 1970. “The least squares problem and pseudo-inverses,” *Comput. J.*, 13, pp. 309–316.

Schenck, H. A. 1968. “Improved integral formulation for acoustic radiation problems,” *The Journal of ASA*, 44, pp. 41–58.

Schenck, H. A. 1993. “Predicting bistatic target scattering from monostatic data,” The 126th Meeting of the ASA, Denver, Colorado, 1993.

Schenck, H. A., G. W. Benthien. 1989. “Numerical solution of Acoustic-structure interaction problems,” NOSC Technical Report 1263.

Tarjan, R. E. 1983. *Data Structures and Network Algorithms,* SIAM, Philadelphia.

Zhang, X., R. H. Byrd, and R. B. Schabel. 1992. “Parallel methods for solving nonlinear block bordered systems of equations,” *SIAM J. Sci. Stat. Comput.*, 13, pp. 841–853.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1994	3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE A DIRECT DECOMPOSITION METHOD FOR THE SOLUTION OF SPARSE LINEAR LEAST SQUARES PROBLEMS		5. FUNDING NUMBERS PE: 0601152N WU: DN302038
6. AUTHOR(S) A. K. Kevorkian		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Command, Control and Ocean Surveillance Center (NCCOSC) RDT&E Division San Diego, CA 92152−5001		8. PERFORMING ORGANIZATION REPORT NUMBER TR 1653
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Independent Research Program Office of Naval Research Arlington, VA 22217−5000		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Authorized for public release; distribution is unlimited.		12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Given a sparse nonsquare system of linear equations Mx = b where M^TM is either dense or full, we present a direct method that generates a least squares solution of the original system by solving a smaller least squares problem. The method accomplishes this decomposition by applying orthogonal transformations to a restructured form of the original system of equations Mx = b. The algorithms derived from the decomposition result are well suited fro both sequential and parallel architecture machines. In a specific Navy signal processing application, the presented algorithm computed on a Sun SPARC 10 workstation a least squares solution of a rank deficient system comprising 703 equations and 592 variables in a number of floating point computations tenfold smaller than a method that does not exploit the sparsity structure of M.

14. SUBJECT TERMS sparse linear least squares			15. NUMBER OF PAGES 41
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAME AS REPORT